An Alternate Soft Computing Approach for Efforts Estimation by Enhancing Constructive Cost Model in Evaluation Method

Brajesh Kumar Singh and A. K. Misra

***Abstract*—Software estimation accuracy is one of the greatest challenges for software developers. Formal effort estimation models, like Constructive Cost Model (COCOMO) are limited by their inability to manage uncertainties and impression surrounding software projects early in the project development cycle. A software effort estimation model which adopts a soft computing technique provides a solution to adjust the uncertain and vague properties of software effort drivers. In this paper, COCOMO is used as algorithmic model and an attempt is being made to validate the soundness of artificial neural network technique using NASA project data. The main objective of this research is to investigate the effect of crisp inputs and soft computing technique on the accuracy of system's output when proposed model applied to the NASA dataset derive the software effort estimates. Proposed model validated by using 85 NASA project dataset. Empirical results show that application of the ANN model for software effort estimates resulted in slightly smaller mean magnitude of relative error (MMRE) and probability of a project having a relative error of less than or equal to 0.25 as compared with results obtained with COCOMO is improved by approximately 17.54%.**

***Index Terms*— Artificial neural network, COCOMO, soft computing, effort estimation, mean magnitude of relative error.**

I. Introduction

Accurate Software development effort estimations are always supposed to be a critical task to both developers and customers. Underestimating the costs of projects may result in exceeding total budget, with underdeveloped functions and poor quality, which may cause delay in projects completion. Overestimating may result in too many additional resources committed to the project, or, during contract bidding, result in losing the contract due to over cost, which can ultimately lead to loss of jobs. So accurate cost estimation is important and software cost estimation involves the determination of effort (usually in person-months), project duration (in calendar time) and cost (in dollars) [1].

Software development effort estimation deals with the prediction of the probable amount of time and cost required to complete the specific development task. Generally, software development effort estimations are based on the prediction of size of software, which is a very difficult task in the sense that estimates obtained at the early stages of development life cycle are inaccurate because not much information of the system is available at initial stage of project development. These estimations are essential for software developers and their companies, because it can provide cost control, delivery accuracy, among many other benefits for them [2].

Now a days, many quantitative models of software cost estimation have been developed. Most of these models are based on the size measure, such as Lines of Code (LOC) and Function Point (FP), obtained from size estimation. It is obvious that the accuracy of size estimation directly impacts the accuracy of cost estimation. Based on this context, new alternative approach in soft computing techniques such as artificial neural networks (ANN) can be a good choice to estimate task effort in software development.

A review of the literature revealed that there are two major types of cost estimation methods Algorithmic and Non algorithmic models as discussed in various papers [3, 4, 5, 6, 7, 8, 9, 10, and 11].

II. About the Problem

A. Problem Statement

Understanding and calculation of models based on historical data are difficult due to inherent complex relationships between the related attributes, are unable to handle categorical data as well as lack of reasoning capabilities Besides, attributes and relationships used to estimate software development effort could change over time and differ for software development environments. In order to address and overcome to these problems, a new model with accurate estimation will be considerable.
We have taken the problem based on algorithmic model i.e. COCOMO into account.

B. Algorithmic Models

Some of the famous algorithmic models are: Boehm's COCOMO'81, II [14], Albrecht's Function Point [12, 13] and Putnam's [14] SLIM. All of these require inputs, accurate estimate of specific attributes, such as Line Of Code (LOC), number of user screen, interfaces and complexity, which are always difficult to acquire during the early stage of software development.

1) The COCOMO

The COCOMO is a regression based software cost estimation model. It was developed by Boehm [13,14] in 1981 and thought to be the most cited, best known and the most plausible [15] of all traditional cost prediction models. COCOMO can be used to calculate the amount of effort and the time schedule for software projects. COCOMO 81 was a stable model on that time. One of the problems with using

Manuscript received April 1, 2012; revised May 10, 2012.

B. K. Singh is research scholar in the Department of Computer Science & Engineering, Motilal Nehru National Institute of Technology, Allahabad, India (e-mail: brajesh1678@yahoo.com).

Dr. A. K. Misra is working as professor in the Department of Computer Science & Engineering, Motilal Nehru National Institute of Technology, Allahabad, India(e-mail: akm@mnnit.ac.in).

COCOMO 81 today is that it does not match the development environment of the late 1990's. Therefore, in 1997 COCOMO II was published and was supposed to solve most of those problems. COCOMO II has three sub models, which are different from those of COCOMO 81.

The limitations of the algorithmic models led to the exploration of the non-algorithmic techniques which are soft computing based.

III. Solution of the Problem

Non-algorithmic models: In 1990's non-algorithmic models have been proposed to project cost estimation. Software researchers have turned their attention to new approaches that are based on soft computing such as artificial neural networks, fuzzy logic models and genetic algorithms. Neural networks are able to generalize from trained data set. A set of training data, a specific learning algorithm makes a set of rules that fit the data and fits previously unseen data in a rational manner (16, 17, 18]. Some of early works show that neural networks are highly applicable to cost estimation [19, 20]. Fuzzy logic offers a powerful linguistic representation to represent imprecision in inputs and outputs, besides providing a more knowledge based approach to model building. Hodgkinson and Garratt represented that estimation by expert judgment was better than all regression based models [21]. However, there is still much uncertainty as to what prediction technique is appropriate to which type of prediction problem [22].

Choosing a suitable technique is a difficult decision that requires the support of a well-defined evaluation scheme to rank each prediction technique as and when it is applied to any prediction problem. In the present study an effective model based on ANN has been proposed to overcome the uncertainly problem and to acquire better results.

IV. Proposed Approach for Solving Problem

In our case we have used feed forward back propagation neural network to solve the problem that will be considered in this section in detail.

A. Dataset Description

We have considered the source data from 93 NASA projects from different centers for projects in years 1971 -1987 Collected by Jairus Hihn, JPL, NASA, Manager SQIP Measurement & Benchmarking Element. Dataset consists of 15 cost drivers, 3 attribute development modes as single input unit, Project Size (in KLOC), Estimated Efforts as output.

B. Proposed Algorithm:

Before applying the proposed algorithm, the neural network is trained in four stages by using feed forward back propagation algorithm for the experimental Data (NASA dataset in the case) viz.

The steps involved in the algorithm are as follows:

(1) Initialization of weights

Step 1:-Initialize weights to small random Values.

Step 2:-While stopping condition is false, do steps 3 – 10.

Step 3:-For each training pair do steps 4-9.

(2)Feed Forward

Step 4:-Each input unit receives the input signal x_i and transmits this signals to all units in the layer above i.e. Hidden units.

Step 5:-Each hidden unit(z_j , j=1,……,p) sums its weighted input signals

$$z_{-inj} = v_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

Next, the BINARY SIGMOIDAL function $z_j = f(z_{in\,j})$ is applied to all units in the layer above i.e. output units.

Step 6:-Each output unit (y_k , k=1,……..,m) sums its weighted input signals,

$$y_{-ink} = w_{ok} + \sum_{j=1}^{n} z_j w_{jk}$$

and applies its BINARY SIGMOIDAL function to calculate the output signals.

$$Y = f(y_{-ink})$$

(3) Back Propagation of errors

Step 7:-Each output unit(y_k , k=1,………,m) receives a target pattern corresponding to an input pattern, error information term is calculated as

$$\delta_k = (t_k - y_k)\, f(y_{-ink})$$

Step 8:-Each hidden unit (z_j , j=1,……...,n) sums its delta inputs from units in the layer above

$$\delta_{-ink} = \sum_{k=1}^{m} \delta_j w_{jk}$$

The error information term is calculated as

$$\delta_j = \delta_{-ink}\, f(z_{-inj})$$

(4) Updation of weight and biases

Step 9:-Each output unit (y_k , k=1,…….,m) updates its bias and weights (j=0,…….,p)

The weight correction term is given by

$$\Delta W_{jk} = \alpha \delta_k z_j$$

And the bias correction term is given by

$$\Delta W_{ok} = \alpha \delta_k$$

Therefore,

$$\Delta W_{jk}(new) = \Delta W_{jk}(old) + \Delta W_{jk}$$

$$\Delta W_{ok}(new) = \Delta W_{ok}(old) + \Delta W_{ok}$$

Each hidden unit (z_j , j=1,………,p) updates its bias and weights (i=0, ………,n).

The weight correction term

$$\Delta V_{ij} = \alpha \delta_j x_i$$

And bias correction term

$$\Delta V_{oj} = \alpha \delta_j$$

Therefore,

$$\Delta V_{ij}(new) = \Delta V_{ij}(old) + \Delta V_{ij}$$

$$\Delta V_{oj}(new) = \Delta V_{oj}(old) + \Delta V_{oj}$$

Step 10:- Test the stopping condition.

V. Result and Discussion

Initially training has been implied by considering 93 datasets of different NASA projects where 17 inputs and 1 output are made available for the purpose. Test data is taken from NASA dataset as input for testing the performance of trained network.

Following Parameters are used for training the network: Model: Back propagation Neural Network

Hidden Layers: 3

Layer 1: 20 Neurons

Layer 2: 15 Neurons

Layer 3: 10 Neurons

Persistence: 200

Learning Rate:

a. Alpha: 0.9

b. Initial eta: 0.3

c. High eta: 0.1

d. Eta decay: 30

e. Low eta: 0.01

Iterations: 50,000

Accuracy: 95.71%

Prevent over training sample: 50%

COCOMO describes 15 cost drivers in which cost drivers are rated on a scale from Very Low to Extra High. Efforts in COCOMO are given as:

Ei =a*(Project Size)b*EAF

Where a, and b depend upon the development mode of the project. Project Size is in KLOC.

EAF is the Effort Adjustment factor based 15 cost drivers.

TABLE I: Constant for Different Project Development Modes

Mode	a	b
Organic	3.2	1.05
Semidetached	3.0	1.12
Embedded	2.8	1.20

VI. Evaluation Method

The value of an effort predictor can be reported many ways including Mean Magnitude of Relative Error (MMRE) and probability of a project having a relative error of less than or equal to L (PRED(L)). MMRE and PRED(L) are the most widely accepted evaluation criteria For evaluating the different software effort estimation.

MMRE and PRED are computed from the relative error, or RE, which is the relative size of the difference between the actual and estimated value of individual effort i :

$$RE_i = (\text{predicted effort}_i - \text{actual effort}_i) / (\text{actual effort}_i)$$

The magnitude of relative error [23] has been calculated by taking the absolute value of that relative error that is,

$$MRE_i = \text{abs}(RE_i)$$

The MRE value is calculated for each observation i of actual and predicted effort. The aggregation of MRE over multiple observations (N) can be achieved through the Mean MRE (MMRE) as follows:

$$MMRE = \frac{1}{N}\sum_{i}^{N} MRE_i$$

A complementary criterion is the prediction at level L, Pred(L) = k/N, where k is the number of observations where MRE is less than or equal to L and N is the total number of observations. Thus, Pred(25) gives the percentage of projects which were predicted with a MRE less than or equal to 0.25.

TABLE II Comparison of Performance between BPN Model and COCOMO

Data set	Model	MMRE	Pred(25%)
NASA 93 Dataset	BPN Model	0.44339	43.53%
	COCOMO model	0.52117298	42.35%
	Improvement(%)	17.54279077	

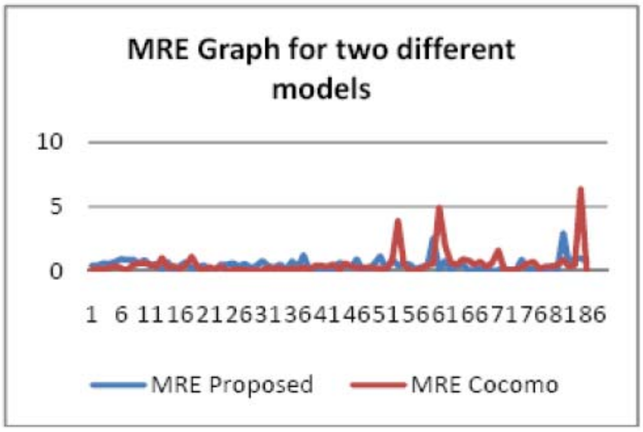

Fig. 1. MRE graph.

Fig. 1 depicts comparison made between 85 results produced by test data for proposed model and corresponding data set for COCOMO as well. We can observe by the Fig.-1 that MRE by proposed model is always kept near to the mean of MRE which shows the accuracy of the model. But in case of COCOMO, there are few spikes with high MRE which show the inconsistency in the evaluation of efforts.

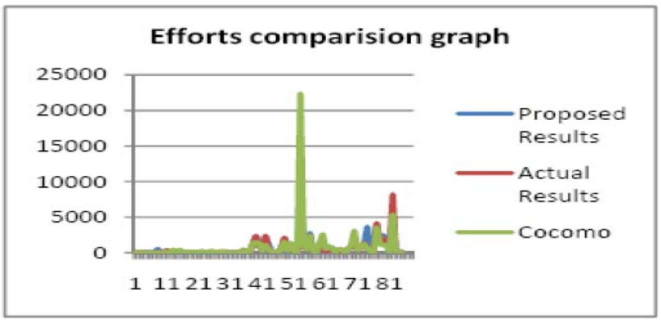

Fig. 2. Effort comparison graph.

Fig. 2 shows the efforts applied with each project during its development. In our case we considered three different sets of efforts that are actual results obtain from NASA datasets, COCOMO results calculating by NASA input data sets, and proposed results are calculated NASA input data sets.

VII. Conclusion

This paper presented a new model for handling imprecision and uncertainty by using the artificial neural network system. This work has further shown by evaluating algorithmic and non algorithmic software effort estimation models that accurate effort estimation is possible. The proposed model showed better software effort estimates in view of the MMRE, Pred(0.25) evaluation criteria as compared to the traditional COCOMO.

It can be observed in Fig.-2 that most of the efforts calculated by proposed models are overlapping with two other methods generated results, which demonstrates that applying proposed model to the software effort estimation is a feasible approach to address the problem of uncertainty and vagueness in software effort drivers. Furthermore, the proposed model presents better estimation accuracy as compared to the NASA dataset. The utilization of soft computing approach for other applications in the software engineering field can also be explored in the future.

References

[1] Parvinder S. Sandhu, Porush Bassi, and Amanpreet Singh Brar, "Software effort estimation using soft computing techniques," *World Academy of Science, Engineering and Technology* pp 46 2008..

[2] Iman Attarzadeh and Siew Hock Ow, "A novel algorithmic cost estimation model based on soft computing technique," *Journal of Computer Science* 6 (2): 117-125, 2010..

[3] B. W. Boehm, "Software engineering economics," Englewood Cliffs, NJ: Prentice-Hall, 1981.

[4] C. E. Walston and C. P. Felix, "A method of programming measurement and estimation," *IBM Systems Journal*, vol. 16, no. 1, pp. 54 73, 1977.

[5] G. N. Parkinson, *Parkinson's Law and Other Studies in Administration*, Houghton-Miffin, Boston, 1957.

[6] L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," IEEE Trans. Soft. Eng., pp. 345-361, July 1978.

[7] J. R. Herd, J.N. Postak, W.E. Russell, and K.R. Steward, "Software cost estimation study: Study results, final technical report," RADCTR77- 220, vol. I, Doty Associates, Inc., Rockville, MD, pp. 1-10, 1977.

[8] R. E. Park, PRICE S, "The calculation within and why," *Proc. of ISPA Tenth Annual Conference, Brighton, England*, pp. 231-240, July 1988.

[9] R. K. D. Black, R. P. Curnow, R. Katz, and M. D. Gray, BCS Software Production Data, Final Technical Report, RADC-TR-77-116, *Boeing Computer Services*, Inc., March, pp. 5-8, 1977.

[10] R. Tausworthe, "Deep space network software cost estimation model," *Jet Propulsion Laboratory Publication* 81-7, pp. 67-78, 1981

[11] W. S. Donelson, "Project planning and control," *Proc. Datamation*, pp. 73- 80, June 1976.

[12] B. Boehm, C. Abts, and S. Chulani, 2000. Software development cost estimation approaches-A survey. Ann. Software Eng., 10: 177-205. DOI: 10.1023/A: 1018991717352.

[13] B. Boehm, 1995. "Cost models for future software life cycle processes: COCOMO 2.0. Ann. Software Eng. 1: 45 60."

[14] L. H. Putnam, 1978. "A general empirical solution to the macro software sizing and estimating problem," IEEE Trans. Software Eng., 4: 345-361. http://portal.acm.org/citation.cfm?id=1313641.

[15] Z. Fei and X. Liu, f-COCOMO: Fuzzy constructive cost model in software engineering, *Proc. IEEE International Conference on Fuzzy Systems*, San Diego, CA., USA.,Mar. 8-12 1992, pp: 331-337. DOI: 10.1109/FUZZY.1992.258637.

[16] K. Srinivasan and D. Fisher, 1995, "Machine learning approaches to estimating software development effort," IEEE Trans. Software Eng., 21: 126-137. DOI: 10.1109/32.345828.

[17] A. Idri, A. Zahi, and A. Abran, "Software cost estimation by fuzzy analogy for web hypermedia applications," *Proc. the International Conference on Software Process and Product Measurement*, (SPPM'06), Cadiz, Spain, 2006, pp: 53-62.

[18] H. Liu and L. Yu, 2005. "Toward integrating feature selection algorithms for classification and clustering," IEEE Trans. Knowl. Data Eng., 17: 491- 502. DOI: 10.1109/TKDE.2005.66.

[19] A. R. Venkatachalam, "Software cost estimation using artificial neural networks," Proc. of the 1993 International Joint Conference on Neural Networks", IEEE Xplore Press, USA. 1993, pp: 987-990. DOI: 10.1109/IJCNN.1993.714077.

[20] S. A. Kumar, Krishna and P. Satsangi, 1994. Fuzzy systems and neural networks in software engineering project management. J. Applied Intel., 4: 31-52. DOI: 10.1007/BF00872054.

[21] A. C. Hodgkinson and P. W. Garratt, "A neurofuzzy cost estimator," *Proc.of the 3rd International Conference on Software Engineering and Applications*, (SEA'99), ePrint, pp: 401- 406. http://eprints.ecs.soton.ac.uk/2659/.

[22] C. J. Burgess and M. Lefley, 2001, Can genetic programming improve software effort estimation? A comparative evaluation. Inform. Software Technol., 43: 863-873. DOI: 10.1016/S0950-5849(01)00192-6.

[23] Z. Xu and T. M. Khoshgoftaar. "Identification of fuzzy models of software cost estimation," 2004.

[24] J. H. Holland, Adoption in Natural and Artificial Systems, Ann Arbor: University of Michigan Press, 1975.

Brajesh Kumar Singh is presently doing Ph.D. from MNNIT, Allahabad, India, under the guidance of Prof. A.K.Misra, Department of Computer Science & Engineering. He is working as Reader in Computer Science & Engineering at FET, RBS College, Agra, India. He has few national and international research papers including IEEE publications. He is the member of international associations in the field of soft computing and software engineering.